

Reducing Buffer Space in Multipath Schemes

M. Yabandeh, S. Zarifzadeh, N. Yazdani and A. Khonsari
Router Laboratory, ECE Faculty, University of Tehran, Tehran, Iran

Abstract— One major drawback of multipath transferring schemes, which is inspired by the usage of different paths with diverse delays, is the emergence of reordering among packets of a flow. This reordering brings some substantial problems (like larger delay and buffer space) to the transport applications. In this paper, we present a novel UDP-based multipath scheme for in-order delivery to the receiver by scheduling of packets among multiple paths. This method imposes the minimum possible delay and a small buffer space on the receiver's application. We theoretically prove the optimality of the proposed method. Finally, through simulation experiments, we show that the performance of our multipath method is comparable with the best-case one-path transmission with aggregated bandwidth.

I. INTRODUCTION

The idea of multipath routing was initially proposed to achieve load balancing [1], fault-tolerance, and more aggregate bandwidth. However, such a scheme causes the packets belonging to the same flow to experience different end-to-end delays. As the first result, this yields the packets to be delivered in a different order with respect to what the source has sent. This reordering results in i) more demanded buffer space; and ii) extra delays for the receiver's application. This makes the application of multipath schemes hardly possible for receiver devices with limited resources, like handheld PDAs. This problem becomes even more serious in wireless ad hoc networks. This is due to the fact that for the sake of reducing the potential interference between paths, the selected paths must be preferably as disjoint as possible. This implies the utilization of more diverse paths which causes more discrepancy between their delays.

Thus far, most of the multipath schemes have tried to bypass the reordering problem by streaming all of the packets belonging to the same flow through a single path [2]-[5]. In fact, they divide the available flows individually between paths. Nevertheless, as reported in [6], a per-packet granularity results in much better performance than the explained per-flow streaming approach.

In [7], Mao et al. extended Realtime Transmission Protocol (RTP) to support the use of multiple paths in Multipath Realtime Transmission Protocol (MRTP), an protocol that relies on UDP as transport. MRTP specifies session establishment, maintenance, and scheduling mechanisms over multiple paths. The other prominent

related works concern mostly the specific side effects of reordering in TCP. In [8], we proposed novel end-to-end streaming mechanisms to transfer packets of a single flow through multiple paths. Our approach schedules transmission of packets over multiple paths in such a way that they are received at the destination in-order. Also, we defined some techniques to reduce fast-retransmit and timeout events in multipath TCP schemes. In this paper, we refine our method in the context of UDP connections and then analyze it, specifically in terms of the required buffer space. Besides, through simulation, the impact of the proposed method on both delay and buffer space metrics is studied in realistic dynamic network conditions.

The remainder of the paper is organized as follows. Section II illustrates the assumed network model. Section III is devoted to the explanation of the proposed method and also its analytical discussion. Our analysis is verified through simulation in section IV. Finally, we conclude the paper in section V.

II. NETWORK MODEL

Assume two network nodes S and D , where node S has some amount of data towards node D . Moreover, assume that there exist n distinct paths from S to D . We use d_i notation to indicate the average one-way delay of path i (i.e. propagation plus queuing and transmission delays). Without loss of generality, suppose that all paths are sorted with respect to their delays, i.e. $d_i < d_j$ ($\forall i, j, i < j$).

As Figure 1 illustrates, the transmission process of a connection in our model is divided into some fixed-size steps. Formally, we define a *transmission step* as a part of transmission process in which all of the paths are participated exactly once (namely, every path carries a continuous set of data in each step). We use *bulk j* to indicate the amount of data which is transmitted over path j in each step. Generally, in each step we stream data bulks of slower paths (i.e. the paths with larger delays) sooner than those of faster ones. As we will show in the next section, operating in this manner allows us to schedule packets among multiple paths in a way that they arrive at the destination in-order. Consider Figure 1 again. Suppose that there exist only two available paths 1 and 2 while path 2 is the slower one. Step i starts with transmitting through path 2. This transmission lasts t_2 seconds. After that, the packets are carried over path 1 for t_1 seconds. With completion of transmission over path 1, the $i+1^{\text{th}}$ step starts with transmission through path 2 again.

It is worth noting that in this model all sending (receiving) times are measured based on the exit (entrance) of packets from (to) IP layer. This justifies the serial transmission scheme which is depicted in Figure 1, despite the existence of multiple network interfaces at the end hosts. The IP layer of the sender is responsible for splitting traffic between the available paths. In other words, during each step which lasts t_{step} seconds, the IP layer sends the incoming packets over path i for t_i seconds which comprise the i^{th} bulk in that step. We define the effective bandwidth of path i (referred to by B_i) as the amount of used bandwidth by the sender through path i , considering the limitation of the intermediary network between S and D . In our model, we utilize paths according to their effective bandwidths, namely:

$$f_i = \frac{t_i}{t_{\text{step}}} = \frac{B_i}{B}, \quad (1)$$

in which f_i is the ratio of bandwidth utilization over path i and B is total effective bandwidth over all paths. For simplicity, hereafter we use the *bandwidth* term to refer to the effective bandwidth.

We assume that the approximate values of paths' delay and effective bandwidth are known for the network (IP) layer. This information could be acquired by a simple extension to some link-state routing protocols such as OSPF (like the ones proposed in [10] and [11]). Another way to realize this assumption is to keep track of on-going/receiving data/ACK packets at the sender to estimate the delay values of paths. Note that in UDP connections, the control packets of RTCP can play the role of ACK packets.

III. THE PROPOSED METHOD

Consider a scenario in which we are going to split packets belonging to a single UDP connection between multiple paths. Clearly, packets which are moved over slower paths will receive later at the destination in comparison with those carried over faster paths. In general, our idea is to schedule the packets among these paths such that the destination receives them in-order. The idea is schematically depicted in Figure 2. The first row in the figure is sequential raw data which is ready for transmission in step i . The second row shows the reordered data which is actually transmitted. Finally, the last row depicts the received data at the destination. As we described before, in each step the sender transmits a continuous bulk of data over every path. In the figure, each data bulk is tagged by the path number which conveys it. The intuition behind our idea is that in each step the data bulks of the slower paths are streamed sooner than those of the faster ones. Because of more delays of path j in comparison with path i ($\forall i, 0 < i < j$), the traffic could be properly scheduled so that the data bulk carried over path j is delivered at the destination after the data bulk of path i . In fact, the main issue is "how to schedule the traffic over path 1 to n in a way that the packets sent on path i ($\forall i, i > 1$) are received exactly after those carried over path $i-1$ ". Since the sender breaks the original order of data, the data should be completely available

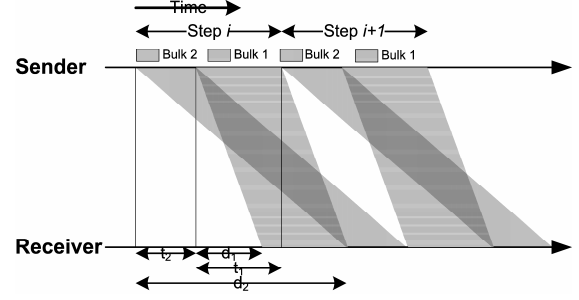


Figure 1. Timeline of sequential transmission steps in the presented model. The transmission time of bulk j lasts t_j seconds and the delay of path k is d_k .

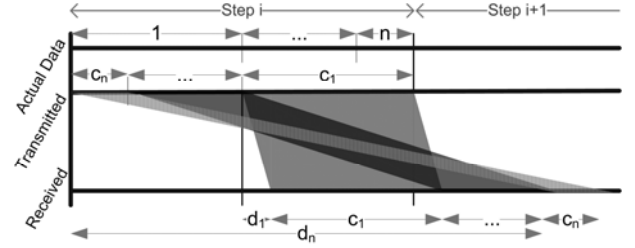


Figure 2. Schematic view of the proposed idea for UDP connections. The method schedules packets at the source to arrive in-order at the destination.

before the start of transmission. Consequently, this approach is not suitable for applications like VoIP and video conferencing which produces raw data instantly. However, it is very useful for some other applications like video/voice on-demand applications which usually have strong senders (e.g. powerful servers), while their receivers are from a broader range, like PDA and PC. Since the limited memory space at the hand-held devices is one of the major factors influencing design options, reducing buffer space at the receiver is much of interest.

To measure the performance of our method, we now define a practical performance metric.

Definition 1. Beginning Pause Time (BPT): the amount of time from the start of transmission (by the sender) that the receiver should pause to get sure that delivering data to the upper layer will not be interrupted.

In other word, *BPT* is the sum of buffer underrun durations for the case that the receiver starts its playing just as the sender begins its transmission. As it can be seen from Figure 2, the *BPT* of our solution is calculated by:

$$BPT = d_1 + \sum_{i=2}^n t_i. \quad (2)$$

To establish the scheduling illustrated in Figure 2, the delay of path i should equal the sum of the following parameters: i) the delay of path 1; ii) the elapsed time for sending traffic through path j for all $j, j < i$ (i.e. after transmitting over path i); and iii) the time required to get sure that the data which will be sent after path i in the same step will arrive at the destination. These issues are summarized in the following equation:

$$d_i = \sum_{j=1}^{i-1} t_j + d_1 + \sum_{j=2}^i t_j = d_1 + 2 \cdot \sum_{j=1}^i t_j - t_i - t_1. \quad (3)$$

This set of independent equations along with those obtained by (1) allow us to deterministically compute the exact values of variables t_1, \dots, t_n . For example, consider the most practical case where $n = 2$, namely we use only two paths between source and destination nodes. As a consequence, (1) and (3) are rewritten as:

$$f_1 = \frac{t_1}{t_1 + t_2}, \quad (4)$$

$$d_2 = d_1 + t_1 + t_2. \quad (5)$$

Suppose that $\Delta d = d_2 - d_1$. Based on above equations, the perfect values of t_1 and t_2 are straightforwardly computed by:

$$t_2 = f_2 \cdot \Delta d, \quad (6)$$

$$t_1 = f_1 \cdot \Delta d. \quad (7)$$

Together with (2), (6), and (7), we can conclude the following BPT for our solution:

$$BPT = d_1 \cdot f_1 + d_2 \cdot f_2. \quad (8)$$

From (8), it is evident that by decreasing the amount of data which will be conveyed through the slower path, the BPT value will be reduced accordingly. However, we should be careful not to decrease f_2 so much that the overhead of packet header becomes intolerable.

Below, we prove that our solution delivers an optimal value for BPT . Assume a general streaming algorithm where the packets are received arbitrarily (i.e. in-order or out-of-order). We consider the best case in which all the received packets are in-order. Suppose an arbitrary point in the axis of time at the receiver. The zero point of the axis represents the start of transmission by the sender. Figure 3 illustrates this.

According to (1), the expected sending rate over path i would be proportional to f_i . Let t be a given time where $t \geq d_i$. Thus, during $[d_i, t]$, the receiver is supplied through path i with u_i units of time for playing, where u_i is calculated by:

$$u_i = \int_{d_i}^t f_i \cdot dt = f_i \cdot (t - d_i). \quad (9)$$

From definition 1, the BPT is equal to the sum of the idle times at the receiver in which it has no data for delivering to the upper layer, namely:

$$BPT_{opt}^n = t - \left[\sum_{i=1}^n f_i \cdot (t - d_i) \right]. \quad (10)$$

When $n = 2$ (i.e. we are using only two paths), the optimal value of BPT would be:

$$BPT_{opt}^2 = t - [f_1 \cdot (t - d_1) + f_2 \cdot (t - d_2)] = f_1 \cdot d_1 + f_2 \cdot d_2. \quad (11)$$

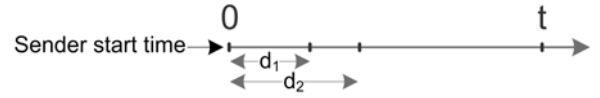


Figure 3. Axis of receiving times from path 1 and 2. The zero point of the axis represents the start of transmission by the sender.

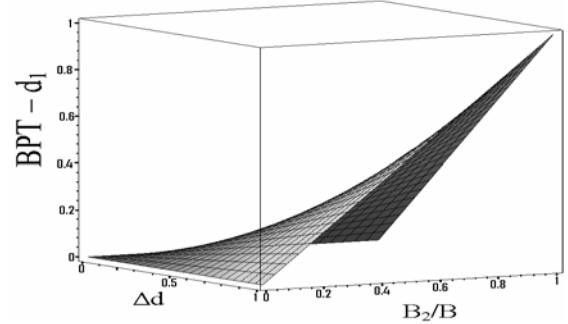


Figure 4. The additional delay imposed by the proposed method on the receiver's application (i.e. $BPT - d_1$) with respect to the delay difference (i.e. Δd) and bandwidth ratio of the slower path (i.e. B_2/B).

This value equals the value of BPT achieved by our solution, as shown in (8). Clearly, the BPT obtained by all other algorithms would be greater than or equal to this value. Consequently, the BPT which the receiver experiences by our algorithm is optimal.

Using Δd instead of $d_2 - d_1$, we can rewrite (8) as:

$$BPT - d_1 = \Delta d \cdot f_2. \quad (12)$$

Remind that d_1 is the inevitable delay of transmission between node A and B , because it is the smallest possible delay between two nodes. Hence, the above equation implies that the additional delay (i.e. $BPT - d_1$) will rise by the increase in the difference between paths' delay and also by the raise in the fraction of data which will be conveyed across the slower path. The analytical results of the BPT with respect to Δd and f_2 are depicted in Figure 4. As the figure shows, the overall delay will increase multiplicatively with the raise in the fraction of data which is conveyed through the slower path.

A. Analyzing the Required Buffer Space

Now, we analyze our method with respect to the amount of buffer space required at the receiver. Based on [13], the delay experienced by each packet over a path y is determined by two parameters: i) the minimum constant delay of path y (represented by D_{min}^y), which can be thought as the propagation delay of that path; and ii) the additional variable delay of path y (denoted by D_{var}^y), which can be seen as the queuing and processing delay of the intermediate routers. Inspired by [13] and [14], we assume that the additional delay of path y (i.e. D_{var}^y) follows an exponential distribution with the average of λ^y . Now, let Δs be the constant interval between transmissions of two consecutive packets at the sender. The value of Δs is determined by the actual throughput of the connection which roughly remains constant despite switching between paths. Also, suppose that S_i^y represents the start time of transmission for the i^{th} packet sent through path y and R_i^y denotes the reception time of

this packet at the receiver. Then, according to the above discussion, the following relations are obtained for every path y , $1 \leq y \leq n$:

$$R_i^y = S_i^y + D_{\min}^y + D_{\text{var}}^y, \quad 1 \leq i \leq \infty. \quad (13)$$

In addition, regarding the fact that in each step all packets of the faster path i are streamed exactly after those of the slower path $i+1$, we can conclude that:

$$S_i^y = S_1^{y''} + \left((i-1) - \sum_{j=1}^{y-1} m_j + \sum_{j=y+1}^n m_j \right) \cdot \Delta s, \quad 1 \leq i \leq m_y. \quad (14)$$

where m_y denotes the number of packets in bulk y . On the other hand, according to (13), we have:

$$\begin{aligned} & P(R_j^y > R_i^{y'}) \\ &= P(S_j^y + D_{\min}^y + D_{\text{var}}^y > S_i^{y'} + D_{\min}^{y'} + D_{\text{var}}^{y'}) \quad (15) \\ &= P(D_{\text{var}}^{y'} - D_{\text{var}}^y < D_{\min}^y - D_{\min}^{y'} + S_j^y - S_i^{y'}) \end{aligned}$$

In order to simplify the calculations, we define the threshold variable α as follows:

$$\alpha = D_{\min}^y - D_{\min}^{y'} + S_j^y - S_i^{y'}. \quad (16)$$

Then, (15) converts into:

$$P(R_j^y > R_i^{y'}) = P(D_{\text{var}}^{y'} - D_{\text{var}}^y < \alpha). \quad (17)$$

Depending on the value of α , we may encounter two different cases:

Case A if $\alpha > 0$:

$$\begin{aligned} P(D_{\text{var}}^{y'} - D_{\text{var}}^y < \alpha) &= \int_{d_{\text{var}}^y=0}^{\infty} \int_{d_{\text{var}}^{y'}=0}^{d_{\text{var}}^y+\alpha} D_{\text{var}}^{y'} \cdot D_{\text{var}}^y \cdot dd_{\text{var}}^{y'} \cdot dd_{\text{var}}^y \\ &= \int_{d_{\text{var}}^y=0}^{\infty} \int_{d_{\text{var}}^{y'}=0}^{d_{\text{var}}^y+\alpha} \frac{1}{\lambda^{y'}} \cdot e^{-\frac{d_{\text{var}}^{y'}}{\lambda^{y'}}} \cdot \frac{1}{\lambda^y} \cdot e^{-\frac{d_{\text{var}}^y}{\lambda^y}} \cdot dd_{\text{var}}^{y'} \cdot dd_{\text{var}}^y \quad (18) \\ &= 1 - \frac{\lambda^{y'}}{\lambda^y + \lambda^{y'}} \cdot e^{-\frac{\alpha}{\lambda^{y'}}} \end{aligned}$$

Case B if $\alpha < 0$:

$$\begin{aligned} P(D_{\text{var}}^{y'} - D_{\text{var}}^y < \alpha) &= \int_{d_{\text{var}}^y=-\alpha}^{\infty} \int_{d_{\text{var}}^{y'}=0}^{d_{\text{var}}^y+\alpha} D_{\text{var}}^{y'} \cdot D_{\text{var}}^y \cdot dd_{\text{var}}^{y'} \cdot dd_{\text{var}}^y \\ &= \int_{d_{\text{var}}^y=-\alpha}^{\infty} \int_{d_{\text{var}}^{y'}=0}^{d_{\text{var}}^y+\alpha} \frac{1}{\lambda^{y'}} \cdot e^{-\frac{d_{\text{var}}^{y'}}{\lambda^{y'}}} \cdot \frac{1}{\lambda^y} \cdot e^{-\frac{d_{\text{var}}^y}{\lambda^y}} \cdot dd_{\text{var}}^{y'} \cdot dd_{\text{var}}^y \quad (19) \\ &= \frac{\lambda^y}{\lambda^y + \lambda^{y'}} \cdot e^{-\frac{\alpha}{\lambda^{y'}}} \end{aligned}$$

Consider i^{th} and j^{th} packet where $j=i+k$ and $k > 0$. We define P_{good}^k as the probability of receipt of j^{th} packet after the i^{th} one. Accordingly, the following probabilities are defined:

$$\begin{aligned} P_{\text{good}}^k &= P(R_j^y > R_i^{y'}), \quad P_{\text{bad}}^k = 1 - P_{\text{good}}^k, \\ P_{\text{relative}}^k &= P_{\text{bad}}^k / P_{\text{good}}^k \quad 1 \leq k \leq \infty \end{aligned} \quad (20)$$

For a given packet i , the next packets will not cause reordering, if all of them will receive after the packet i . The probability of this event is:

$$P_{\text{allgood}}^i = \prod_{k=1}^{\infty} P_{\text{good}}^k \quad 1 \leq i \leq \infty, \quad (21)$$

This yields that the maximum buffer size equals 1 for successful (in-order) delivering of the i^{th} packet to the upper layer. Generally, the probability that the maximum buffer size for successful delivery of the i^{th} packet reaches l , is the probability of reordering of $l-1$ individual of the packets sent after the i^{th} packet. This probability (referred to by $P_{\text{maxbuffer}}^{l,i}$) can be calculates as follows:

$$P_{\text{maxbuffer}}^{l,i} = \sum_{i_1=0}^{\infty} \cdots \sum_{i_{l-1}=i_{l-1}+1}^{\infty} \left(P_{\text{allgood}}^i \cdot \prod_{k=1}^l P_{\text{relative}}^{i_k} \right), \quad (22)$$

Clearly, the probability that the required buffer space never exceed l throughout the whole communication, i.e. $P_{\text{maxbuffer}}^l$, is the maximum of $P_{\text{maxbuffer}}^{l,i}$ over all given values of i , namely:

$$P_{\text{maxbuffer}}^l = \max_i P_{\text{maxbuffer}}^{l,i} \quad 1 \leq i \leq \infty, \quad (23)$$

In our network model, the lifetime of a connection is comprised from several fixed-size steps. So, we just evaluate the corresponding probabilities of packets inside a given step. Note that in the special case of one-path scenario, the probability of (22) remains the same for all given values of i . For practical applications, the maximum length of buffer which will be realized in 95% of situations is of interest. Hence, the required buffer space at receivers, l , will be minimum value of k for which $P_{\text{maxbuffer}}^{l,k} < 0.05$. Table 1 presents the obtained analytical results for our multipath method.

TABLE I. BUFFER LENGTH AT RECEIVER, ACHIVED ANALYTICALLY FOR TRANSFERRING A FILE OF SIZE 100KB; THE ROWS AND COLUMNS REPRESENT THE RATIO OF BANDWIDTH AND DELAY OF PATH 2 TO PATH 1, RESPECTIVELY.

B_2/B_1		0.48	0.86	1.25	1.64
d_2/d_1					
2		2	3	3	3
2.5		3	4	4	4
3		4	4	4	5
3.5		5	5	5	5
4		5	6	6	6

IV. PERFORMANCE EVALUATION

In this section, we first present the simulation model used to evaluate the performance of the proposed method and then provide the experimental results.

A. Simulation Model

For the purpose of simulation, we implemented RTP/RTCP protocol in Java. The complete code of our simulator can be found in [12]. In our simulations, two threads act as a client and a server. A file will be transferred from the server to the client through a UDP connection. The implemented code simulates the common characteristics and limitations of real networks like dynamically changing latency of links and also drop behavior when the traffic load exceeds the link capacity. Moreover, it enables the IP layer to split the outgoing traffic among multiple distinct paths.

We suppose that path 1 has a smaller delay than other path. The interface bandwidth of two paths is the same (i.e. 2Mbps) and the size of all data packets is fixed to 0.5KB. Also, the bandwidths of path 1 and 2 are limited to B_1 and B_2 , respectively. The configurable parameters of our experiments are i) the relative latency and ii) the relative bandwidth of the paths.

Overall, three different UDP-based methods are simulated in our experiments: i) the usual one-path UDP with aggregated bandwidth of $B_1 + B_2$; ii) the simple multipath UDP which simply divides the traffic between two paths according to their bandwidth ratio; and iii) our enhanced multipath whose transmission parameters are set based on (6) and (7). Clearly, the results of the one-path scenario with aggregated bandwidth imply the optimal values which could be obtained by any perfect multipath method. In the simple multipath scenario, the outgoing packets are divided at the IP layer between two available paths according to the ratio of their bandwidths (i.e. in each point of the time, path 1 has carried $B_1 / (B_1 + B_2)$ portion of the outgoing data). We measure the BPT parameter and the consumed buffer space at receiver to compare performance of these methods.

In all of the following experiments, we assume that the average latency and the bandwidth of path 1 are fixed to 0.05s and 128Kbps, respectively. Also, all diagrams are plotted based on the relative latency and bandwidth of path 2 with respect to those of path 1.

B. Experimental Results

Figure 5 shows the experimental results of BPT obtained by different methods for transferring a file of size 100KB. To give the reader better insight, the figure also presents the analytical result of BPT in our approach, calculated based on (8). As we proved in Section III, this result shows the minimum value of BPT that can be obtained by any multipath method. According to the figure, the one-path method completely outperforms two other multipath approaches, especially when d_2 / d_1 is high. However, our enhanced multipath method significantly improves the BPT , compared with simple multipath approach. As much as the delay ratio of two paths becomes higher, this improvement becomes more apparent. Also, the simulation results of our approach are on average 20% above the analytical ones, which is mainly due to the existence of packet delay variance and drop behavior in our experiments.

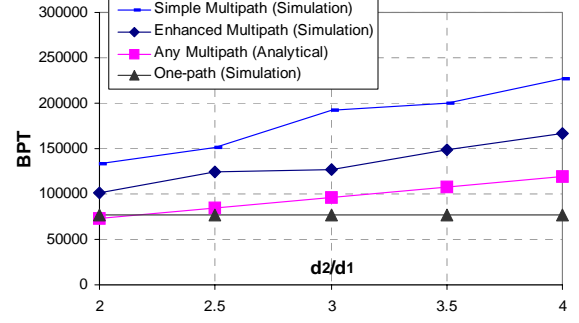


Figure 5. BPT achieved by UDP methods for transferring a file of size 100KB.

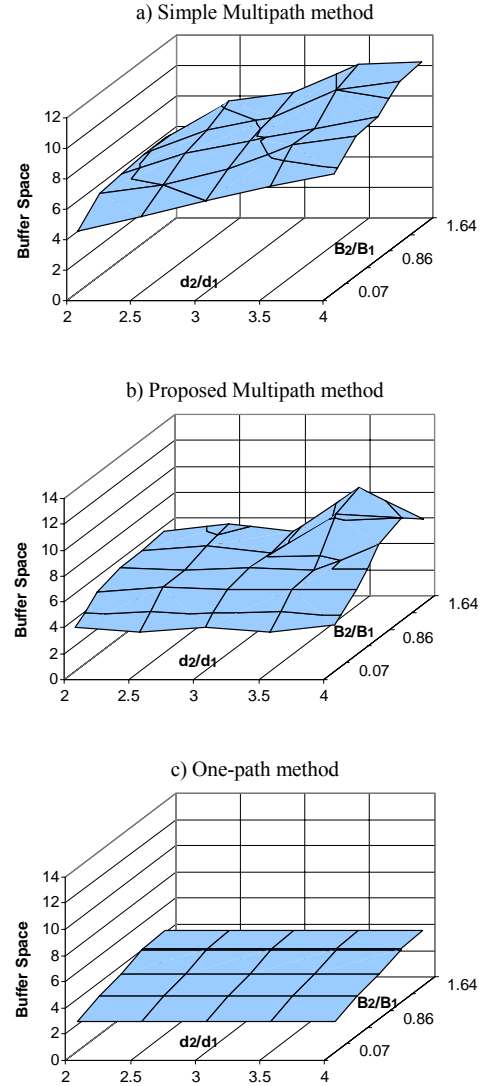


Figure 6. The measured buffer level of three methods.

With the assumption that the receiver's application starts its playing with the optimal measured BPT , the consumed buffer space at the receiver is depicted in Figure 6. As illustrated in the figure, the consumed buffer space generally increases with the raise in the effective bandwidth ratio. In the case of simple multipath scenario, the buffer level grows drastically with the increase in the

delay of the slower path. Although we can observe this growth in our approach too, but its variation has been well controlled now. One can easily comprehend the similarity between the analytical results, presented in the previous section, and these experimental ones. For instance, both results increase as the ratio of B_2 / B_1 becomes higher.

To better demonstrate the effect of our proposed method with respect to buffer space, a 2-D snapshot of Figure 6 is rendered in Figure 7, comparing the results of all three methods. As the figure shows, the consumed buffer level of our method is close to the inevitable level implied by the one-path method. Also, the level of buffer space remains steady throughout the increase in the delay ratio of path 2 to path 1. This is in spite of the simple multipath method that the buffer level raises drastically with the increase in the delay ratio.

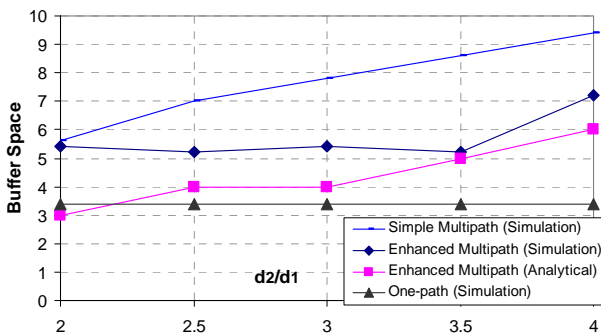


Figure 7. The measured buffer level of three methods for B_2 / B_1 equals to 0.86 through different delay ratios.

V. CONCLUSION

The main problem of multipath schemes is the difference between the delays of selected paths which causes reordering among packets of the same flow. This imposes a larger buffer space and extra delay on receiver's application. Multipath methods in both wired and wireless networks prefer more diverse paths and this unfortunately intensifies the problem. In this paper, a novel streaming approach for handling the reordering problem in end-to-end multipath schemes was presented. Our method schedules packets at the sender among multiple paths in a way that they arrive at the receiver in-order. It was proven that our proposal imposes the minimum possible delay on the receiver's application to start after the sender begins its transmission. In addition, our method reduces the need for large buffer spaces at the receiver. Simulation results also confirm the efficiency of the proposed method in terms of delay and buffer space. Interestingly, the consumed buffer is close to the buffer required by the optimal one-path method (with aggregate bandwidth) and also is relatively indifferent to changes in delay ratios.

REFERENCES

- [1] R. Rom, I. Cidon, and Y. Shavitt, "Analysis of multi-path routing," *IEEE/ACM Transactions on Networking*, vol. 7, issue 6, pp. 885-896, December 1999.
- [2] S. J. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *Proc. of IEEE International Conference on Communications (ICC)*, Helsinki, Finland, pp. 3201-3205, 2001.
- [3] W. Wei and A. Zakhori, "Robust multipath source routing protocol (RMPSR) for video communication over wireless ad hoc

- networks," in *Proc. of IEEE Int'l Conference on Multimedia and Expo (ICME 2004)*, Taipei, 2004.
- [4] M. K. Marina and S. R. Das, "On-demand multipath distance vector routing in ad hoc networks," in *Proc. of International Conference for Network Protocols (ICNP)*, 2001.
- [5] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, "A framework for reliable routing in mobile ad hoc networks," in *Proc. of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, San Francisco, CA, USA, 2003.
- [6] R. Krishnan and J. A. Silvester, "Choice of allocation granularity in multipath source routing schemes," in *Proc. of 12th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '93)*, San Francisco, CA, vol. 1, pp. 322-329, March 1993.
- [7] S. Mao, D. Bushmitch, S. Narayanan, and S. S. Panwar, "MRTP: a multi-flow realtime transport protocol for ad hoc networks," in *Proc. of IEEE Vehicular Technology Conference*, October 2003.
- [8] M. Yabandeh, S. Zarifzadeh, and N. Yazdani, "Improving Performance of Transport Protocols in Multipath Transferring Schemes," to be appear in *Elsevier Computer Communications*.
- [9] S. Norden, M. M. Buddhikot, M. Waldvogel, and S. Suri, "Routing bandwidth guaranteed paths with restoration in label switched networks," in *Proc. of IEEE International Conference on Network Protocols (ICNP 2001)*, Riverside, CA, USA, pp. 71-79, November 2001.
- [10] G. Apostolopoulos, et al., "QoS routing mechanisms and OSPF extensions," *IETF RFC 2676*, 1999.
- [11] W. H. Liao, Y. C. Tseng, S. L. Wang, and J. P. Sheu, "A multipath QoS routing protocol in a wireless mobile ad hoc network," *IEEE International Conference On Networking*, France, part II, pp. 158-167, July 2001.
- [12] <http://web.ut.ac.ir/routerlab/members/yabandeh/tcp-sim.zip>.
- [13] C. Demichelis and P. Chimento, "IP packet delay variation metric for IP performance metrics (IPPM)," *IETF RFC 3393*, 2002.
- [14] A. Corlett, D. Pullin, and S. Sargood, "Statistics of one-way internet packet delay," *IETF Internet-Draft*, draft-corlett-statistics-of-packet-delays-00.txt, Aug. 2002.